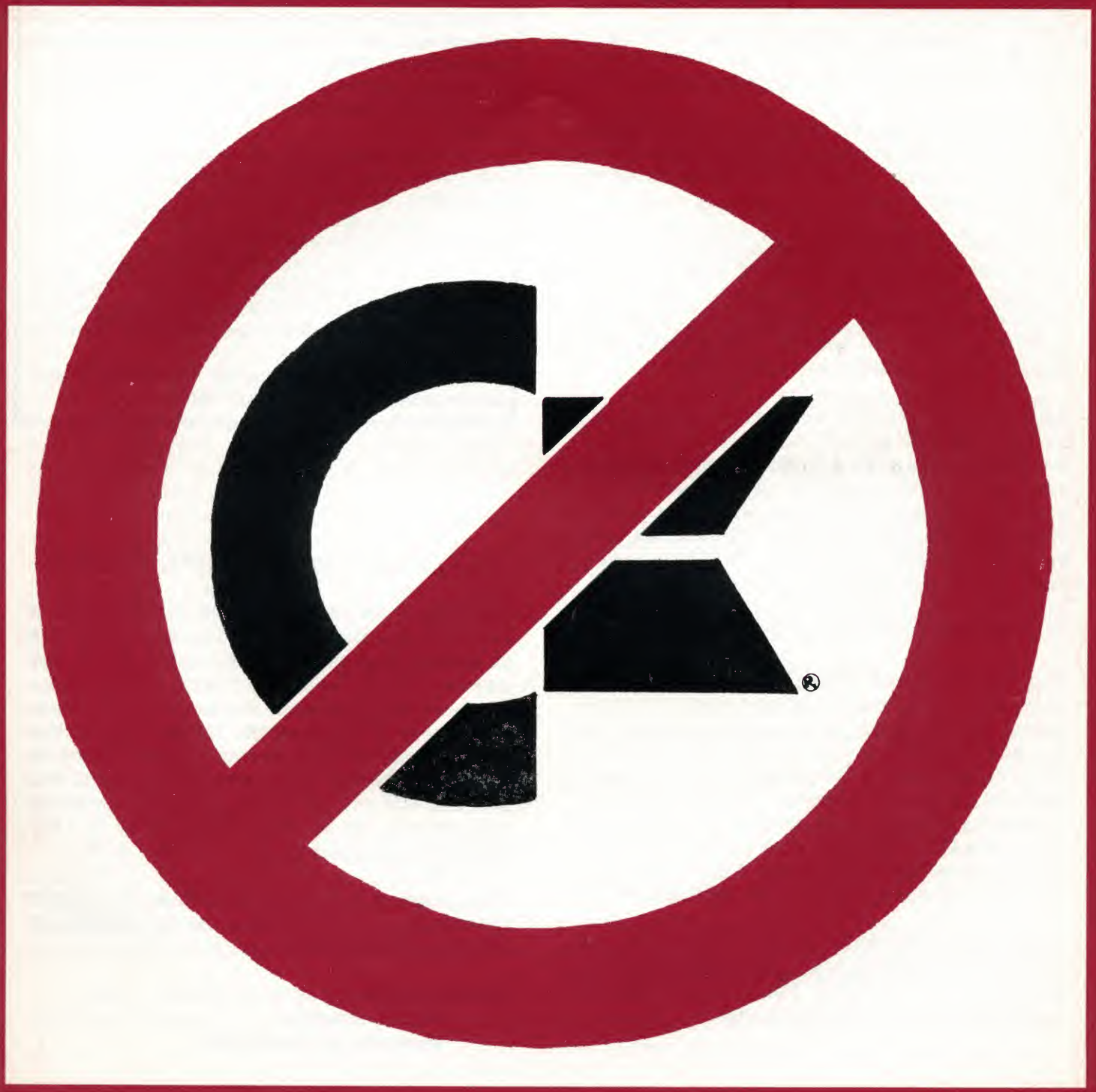
$2.25
August, 1984
Vol. 4 NO. 8

M.A.C.E. JOURNAL

"Devoted Exclusively To The Atari Computer User"



Published by the Michigan Atari Computer Enthusiasts

Presidential Ramblings

Some idle thoughts about our new owners...

WHY SELL ATARI NOW? Another dumb move by Warner Communications, in this writer's opinion. Sure, half-billion dollar losses are no picnic, but that's one hell of a tax write off. And with Atari's soon return to profitability, that makes the move even more strange. They've been watching "MTV" too long! Do all of us Atarians now own Ataridores? If you buy that one, I've got some Michael Jackson concert tickets for you!

ELECTIONS are coming real soon. If you think you can do a better job than I (or any of the officers) please let me or past-president Marshall Dubin know.

TARICON '84 Plans are progressing, slow but sure. Pre-registration for MACE members will be held tonight and at the August meeting. Tickets will cost a minimal $2.00. We anticipate 20 to 30 booths from ATARI and OSS down to your local software dealer.

Adult volunteers will be needed. This is an official MACE activity, and for it to be successful, we need you to volunteer your services for the weekend. Set-up will take place on Friday the 24th, with the show scheduled the 25th and 26th. Tear-down will be after 6:00 PM on the 26th. All this takes place the weekend before Labor Day, so nobody's vacations will be messed up. VOLUNTEER YOUR SERVICES TONIGHT!

The official MACE Road Trip took place on July 4th. Didn't get anyone else to come along, so it proved to be a nice vacation in Canada for yours truly and his family. They've got some real problems with Atari north of the border. See me after the meeting, at Tequila Willie's (the un-official watering hole of MACE), and I'll clue you in on them. My thanks go out to Steve and Chris Gauthier for their warm hospitality (and cold Labatts!).

That's about all this month. Please make yourself available to help with TARICON if at all possible! Call the MACE HOTLINE to volunteer or to get the latest info.

ONE MOMENT PLEASE!

Won't you take a moment right now and take a look at the mailing lable? You know, the one on the back cover. Notice that date in the upper right-hand corner? It's your membership expiration date. If it is close, then renew it now, either by mail or at the next general meeting, to avoid interruption of your membership. Also look at the name and address for any typo's and let me know so corrections can be made. Your help in this area will be appreciated!

If you plan on a change in address, try to get the info to us so our files can be updated at the earliest possible date.

Thanks alot folks!

Paul R. Wheeler
Membership Coordinator

OSS BASIC XL

Reviewed by Mike Portuesi

Good old Atari BASIC. Have you ever wished it could do just a little more? Did you ever want your programs to run just a little faster? Did you ever look at programs for other computers in magazines and wonder just how in the world you were going to translate them to the Atari? Did you ever want to write a program using (sigh of envy) string arrays? Well, all your wishes have been answered. Optimized Systems Software's (OSS) BASIC XL is everything Atari BASIC isn't, and more.

BASIC XL comes to you in the form of a bright orange cartridge enclosed in a cheerful yellow binder. The binder contains a reference manual and the tutorial "Thirty Days to Understanding BASIC XL," a guide for the beginning user. In all, I found the documentation clear and well-explained. The reference section of the manual reads very much like an expanded version of the Atari BASIC Reference Manual. This should come as no surprise, since BASIC XL was written by the same company responsible for Atari BASIC. In a few instances, I found some minor errors in the manual, both in the reference section and the tutorial. I sincerely hope OSS intends to correct these errors as soon as possible.

First of all, I would like to clear up any misconceptions the title may imply. BASIC XL will run on any Atari computer, not just the new XL models. It is upwardly compatible with Atari BASIC; this means it will run normal BASIC programs, but programs written using special BASIC XL features will not work with Atari BASIC. And what features it has! BASIC XL has the following commands not present in Atari BASIC:

* BGET, BPUT - Allows you to do direct binary I/O. This was possible in BASIC only through a series of POKES and a machine language call. Now operations such as saving hi-res screens to disk are almost trivial.

* DEL - Delete a range of line numbers from your program.

* DIR, ERASE, RENAME, PROTECT, UNPROTECT - Say good-bye to the DOS menu. These important disk functions are only a command away.

* DPOKE - Performs a word-oriented (16 bit) POKE into two successive memory locations. Ever see program segments that ran like this?

```
100 HI=INT(MEM/256):LO=MEM-256*HI
110 POKE ADDR,LO:POKE ADDR+1,HI
```

In BASIC XL, that would look like this:

```
100 DPOKE ADDR,MEM
```

DPEEK is the counterpart to DPOKE, and works in much the same manner.

* FAST - Speeds up program execution by doing a precompile of the program in memory. More on this later.

* IF...ELSE...ENDIF - There is is more here than just IF...THEN...ELSE. More on this later.

* INPUT with a prompt following, as in this example:

```
100 INPUT "NAME YOUR POISON",A$
```

* String concatenation - In Microsoft BASIC, strings may be added together like this:

```
100 A$="HI"
110 B$="THERE"
120 C$=A$+" "+B$
130 PRINT C$
```

When RUN, the program would produce:

```
HI THERE
```

In BASIC XL, line 120 would read:

```
120 C$=A$," ",B$
```

* LOMEM - Change the system low memory pointer to save space for assembly language routines, custom character sets, etc.

* LVAR - Lists all variable names and the line numbers they can be found to any output

device (screen, printer or disk file.)

* MOVE - Perform block moves of memory at machine language speed. Blocks may be moved either up or down in memory. No address checks are made, so watch out!

* PMGRAPHICS, PMCLR, PMCOLOR, PMMOVE, PMWIDTH, MOVE, MISSILE, BUMP - A complete set of player-missile commands and functions. A system memory map is now no longer a necessity to use player-missile graphics from BASIC.

* NUM - Auto line numbering. A convenience to save time when entering programs.

* PRINT USING - Allows you to produce neat, formatted output without using subroutines to justify numbers and add trailing zeroes before printing.

* RENUM - When you're finished with your program, why not renumber it so people can't tell you spent the past week or so hacking away at it?

* RGET, RPUT - Perform I/O using records of mixed strings and numeric variables. Most useful in file processing.

* SET - A command that gives you control over various BASIC XL system settings. Do you hate the question mark an INPUT statement produces as much as I do? Change it to a greater-than sign (">") as follows:

```
SET 2,ASC(">")
```

SET allows you to change other system settings as well. The BREAK key can be enabled, disabled, or return an error. The spacing for tab stops can be changed. Missiles can either wraparound from the top of the screen to the bottom, or roll off the edge. Missiles can also be grouped into a fifth player. FOR...NEXT loops can be set to execute at least once, or zero times. You can even decide whether the USR function should push a parameter count on the 6502 stack.

* TAB - Another feature missing from Atari BASIC that was usually done using various POKES to cryptic memory locations.

* TRACE, TRACEOFF - Prints the line number currently being executed on the screen. This can be a timesaver when debugging.

* WHILE...ENDWHILE - A control structure that allows you to perform a group of statements until a certain condition exists. More on this will follow as well.

Is that enough for you? No? Well, then let's take a look at the BASIC XL function library:

* ERR - ERR(0) will return the latest error number. ERR(1) will return the line at which the error occured. This can be useful for error-handling routines.

* HEX$ - Converts an integer number up to 65535 to a hexadecimal string. BASIC XL can also operate and do calculations with hexadecimal constants, as in the following example:

```
100 PRINT HEX$(USR($680,$3FFA,$2972))
```

Typing PRINT $FCB5 would yield a result of 64693. BASIC XL eliminates the need for hexadecimal-decimal conversion tables and programs.

* FIND - Finds the location of a substring in a larger string, no matter how large the string is.

* HSTICK, VSTICK - Returns -1, 0, or +1 for the joystick both horizontally and vertically. Useful in games and etch-a-sketch type programs.

* LEFT$, RIGHT$, and MID$ - Microsoft BASIC, eat your heart out. These functions return substrings of a larger string. Of course, you can still do things they way they are done in Atari BASIC (e.g. A$(M,N)).

* PEN - PEN(0) returns horizontal coordinates of the light pen. PEN(1) returns vertical coordinates.

* PMADR - Finds the address in memory where a given player-missile image is stored.

* RANDOM - Returns a random integer between any two numbers. For a random number between 1 and 100, try:

```
100 R=RANDOM(1,100)
```

* SYS - Returns information on various BASIC XL system settings (the same ones you can change with the SET command).

And let us not forget some of the incidental benefits of BASIC XL:

* String Arrays - This is best explained with an example. The command:

```
DIM A$(20,40)
```

DIMensions an array of 20 elements, each 40 characters long. Referencing an element of the array would look like this:

```
PRINT A$(I;)
```

This references element I in the array. A substring of that element would look like:

```
PRINT A$(I;M,N)
```

* Auto-dimension of strings - BASIC XL will automatically DIMension a string that has not been previously DIMensioned to a length of 40 characters. The auto DIMension length can be changed with the SET command.

* Full English error messages - Never run to the reference manual to find out what "ERROR- 13 IN LINE 32015" means again.

* Formatted listings - BASIC XL will print listings in upper and lower case, indenting FOR...NEXT and WHILE...ENDWHILE loops, and IF...ELSE...ENDIF structures as well.

* Program entry in upper/lower case and inverse video - BASIC XL will automatically convert all characters except those in quotes to upper case normal video, so you can enter programs in lower case.

* Boolean arithmetic - Operators have now been provided for bitwise AND, OR and EOR of two numbers. Machine language programmers know what I'm talking about; others can be content to know they exist.

* FAST program execution - Whenever Atari BASIC encounters a GOTO, GOSUB, or NEXT statement, it has to search for the line to transfer control to, starting at the very beginning of the program and continuing until the line is found. Obviously, the longer your program is, the slower subroutines and such near the end will run. When BASIC XL sees the FAST command, it looks at the program in memory. It replaces the line numbers after GOTOs and GOSUBs with the actual memory addresses of those lines. As a result, programs run much faster. The speed improvement depends on the program. OSS claims programs can run up to four times faster; my experience is about twice normal speed or less. Note that programs that perform a large number of calculations will not benefit very much from the FAST command. Others may benefit very, very much. I have one game in BASIC that is actually too fast to play.

* IF...ELSE...ENDIF capability - Other BASICS have IF..THEN...ELSE. BASIC XL has more. Try this example (from the BASIC XL manual) on for size:

```
200 IF A>100:PRINT "TOO BIG"
210 A=100
220 ELSE:PRINT "A-OK"
230 ENDIF
```

Look at line 200 carefully. Note that there is no THEN statement--only a colon. When using IF...ELSE...ENDIF in BASIC XL, ALL statements (even those on multiple lines) between the following colon and the ELSE statement are executed if the statement is true, and ALL statements between the ELSE and ENDIF are executed if the statement is false. If you didn't quite catch the implications of all this, here's another example from the manual:

```
100 IF A>B : REM SO FAR A IS BIGGER
110   IF A>C : PRINT "A BIGGEST"
120   ELSE : PRINT "C BIGGEST"
130   ENDIF
140 ELSE
150   IF B>C : PRINT "B BIGGEST"
160   ELSE : PRINT "C BIGGEST"
170   ENDIF
180 ENDIF
```

If you like the way that example reads, you will definitely have a fun time with BASIC XL.

* WHILE...ENDWHILE - Another powerful control structure, WHILE...ENDWHILE allows you to set up a loop that executes only as long as some condition holds true. Again, an example:

```
100 WHILE PEEK(53279)<>6:ENDWHILE
```

This line will patiently wait until the START key is pressed. As long as the condition holds true, the loop will execute. When the START key is pressed, the condition becomes false, and BASIC XL stops executing the loop. Contrast this with the equivalent BASIC code:

```
100 IF PEEK(53279)<>6 THEN 100
```

Using the IF...ELSE...ENDIF and WHILE...ENDWHILE capabilities of BASIC XL, it is possible to write structured BASIC programs. GOTO statements are no longer needed.

At this point, you may be wondering "How much extra memory are all these nifty features going to take up in my Atari?" The answer: not a single byte more than Atari BASIC. That's right, the BASIC XL cartridge takes up only 8K in an Atari on power-up. With OS/A+ version 2.1 on my Atari 800, BASIC XL reports 31502 free bytes of memory when PRINT FRE(0) is typed. With Atari DOS 2.0, 32274 bytes of memory are free. These figures are the same for Atari BASIC--check them out yourself. The designers at OSS invented a special bank-switching system that keeps only an 8K portion of the cartridge in memory at one time, while the other 8K is hidden from the machine. The result? With BASIC XL, you can have your cake and eat it too. But that's not all! When used with DOS XL, another OSS product, another 5K is opened for your use. DOS XL was specifically designed for use with BASIC XL, and locates itself in the RAM that is occupied by the cartridge. When DOS is needed, the BASIC XL cartridge ROM is turned completely off, exposing the RAM underneath. When used with DOS XL, about 37K is free for programming in BASIC XL.

Now, for some of my complaints about BASIC XL. My first complaint is the cartridge itself, which has tin connector tabs rather than gold ones. I have worked with early Ataris that had tin-plated connector tabs on their memory boards, and I know what a pain they can be. The boards usually had to be cleaned weekly to avoid system crashes that destroyed hours of work. Obviously, a product of this quality deserves a better cartridge. My second complaint concerns some possible bugs in the program. I bought BASIC XL after seeing it run on a friend's machine a few weeks earlier. When I plugged my BASIC XL cartridge into my Atari, I discovered I had version 1.02. My friend's version 1.0 BASIC XL was purchased only a month earlier. I noted that his cartridge had some compatibiltiy problems with some Atari BASIC programs in my collection. One program stopped running with a LINE NOT FOUND error. This error occurs if you try to GOTO a line that doesn't exist in your program. The problem was that the line where the error occured was a DRAWTO statement! A few programs (such as the Scriptor Word Processor from COMPUTE! Magazine, April 1983 and some games from COMPUTE!) wouldn't run at all. The revision 1.02 cartridge I received did not have these problems (except Scriptor--it still doesn't work), but one cannot be sure if other problems exist. It seems safe to say that most Atari BASIC programs will run properly under BASIC XL, and the ones that don't have exotic machine language routines and other "tricks" in them that make problems for BASIC XL. If I find a program that doesn't work in BASIC XL, I simply run it under Atari BASIC. The original bugs of Atari BASIC have been fixed, however, and you need not worry about a system lock-up while editing a program.

The only real competition for BASIC XL is Atari Microsoft BASIC II. This combination cartridge/diskette package does have a few features that BASIC XL doesn't offer (such as integer variables and user-defined functions of the type DEF FN), and it does offer faster math computation than BASIC XL (BASIC XL uses the slow Atari floating point ROMs). But when you consider that BASIC XL is compatible with Atari BASIC, has syntax-checking, takes up 8K in your Atari as opposed to 18K, offers a FAST command, and comes all on one cartridge instead of a cartridge and a disk, BASIC XL wins hands down. In short, BASIC XL is the first BASIC worthy of the computer it was written for--the Atari.

Have we got software? Do we offer great deals? YES!

Save 40-60% at P.C. EXPRESS's $20.00 Sale!

Choose any title below for only $20.00! Tremendous savings off our regular prices!

You will receive a free gift with every order!

Buy any 5 items and get the 6th one free!

Choose from these selected titles: (D=Disk C=Cassette R=Cartridge)

Preppie	D	Canyon Climber	D	Jeepers Creepers			
Preppie II	DC	Pooyan		Name That Song			
Analog Adventure	D	Tumble Bugs		Starbase Hyperion			
Race in Space	DC	O'Riley's Mine		Wizard of Wor	D		
Shooting Gallery	DC	Moon Shuttle		Gorf	R		
Ghost Hunter	DC	Sands of Egypt	D	Graphics Machine			
Microsoft Basic		Alien Garden		Tricky Tutorial 1	D		
Protector II		Platter Mania		Tricky Tutorial 2	D		
Protector	D	Monster Maze		Tricky Tutorial 3	DC		
Ft. Apocalypse		Keys to Archeron	DC	Tricky Tutorial 4	C		
Shamus	DC	Danger Drindisti	C	Tricky Tutorial 5	DC		
Figure Fun	C	Star Warrior		Tricky Tutorial 6	C		
Super Cubes, Ship	C	Hockey at 102	DC	Picnic Paranoia	DC		
Hickory Dickory D	C	Rosen's Brigade		Stellar 7			
Humpty Dumpty J&J	C	Shuttle Intercept	D	Sky Rescue			
Lords of Karma	C	GO	C	Snake Byte	D		
N Atl Convoy Raid	C	Sentinel One	DC	Space Eggs	D		
Planet Miners	C	Crypts of Terror	D	Cyclod	D		
Controller	D	Intruder	DC	Sneakers	D		
Moon Patrol	C	Math Pack	C	Worm War I	R		
Ghostly Manor		Alien Numbers	C	Turmoil	R		
Stellar Shuttle	D	Alien Swarm	DC	Twerps	D		
Genetic Drift	DC	Action Quest	DC	Micro Mother Goose			
Labyrith	D	Journey to Planets					
Seafox	D	Slime	D				
David Night Magic	D	Dodge Racer	D				
Gumball		Survivor					
A.E.	D	Bumper Ball	DC				
Serpentine	DR	Space Ace	DC				
Raster Blaster		Hot Lips	DC				
Kick Back	R	Peeping Tom	D				
EMI Compute 4/Rev	C	Wingman	D				
Kid Grid	D	Chopper Rescue	DC				
Iridis #1	DC	The Jar Game					
Iridis #2	DC	Gulp					
Aztec Challenge	DC	Battling Bugs					
Crypt of Plumbous	DC	Alien Encounter					
Spider Invasion	DC	Frenzy					
Galactic Avenger	DC	Golf Classic					
Tilt	C	Ric-O-Chet	D				
Dominoes & Cribb	C	Titan Empire					
Pool,Snooker,Bill	C	Mouse Attack	D				
Cosmic Tunnels	D	Jaw Breaker	DC				
Spectre		Mission Asteroid					
Vortex		Claim Jumper	DCR				
Pacific Coast Hwy	D	Frogger					
Cytron Masters		Galactic Gladiators					
Black Forest		Nautilus	DC				
Adventure on a Boat		Chicken	CR				

Don't miss these deals on great APX software!

Text Analysis Rev. II	D	Word Maker	D
RPN Calculator Simulat	D	Weekly Planner	D
Video Math Flash Cards	DC	Wizard's Gold	D
Pro Bowling	D	Escape to Equatus	D
Number Blast	D	Isopeth Map Making	D
Frog Master	D	Dsembler	D
My First Alphabet	D	Meltdown	C
Cubbyholes	D	Dice Poker	C
Salmon Run	D	Air-Raid	C
The Market Place	D	Lookahead	C
Downhill	D	Basic / XA	C
Outlaw Howitzer	D	Mankala	C
Flags of Europe	D	Block 'em	C
Dog Daze	DC	Domination	C
Pushover	D	Chameleon CRT Term	C
Caterpiggle	D	Blackjack Casino	C
Sketchpad	D	Seven Card Stud	C
Tact Trek	D	Babel	C
Mugwump	D	Yahtman	C
Quiz Master	D	Memory Match	C

Don't miss these deals on great APX software!

Block Buster	D	Music Player	D
T:Text Display Device	DC	Centurion	D
Jax-O	D	Cosmatic Atari Dev Pak	D
Rabbotz	D	Counter	D
Family Budget	DC	Quarxon	DC
Personal Fitness Prog.	D	Insomnia Rev. 1.1	D
Codecracker	D	Checker King	DC
Fingerspelling	DC	Reversi II	D
Math Mission	D	Fog Index	D
Snark Hunt	D	I'm Different	D
Musical Computer	D	Going to the Dogs	D
Pushky	D		

NOTE: Quantity is limited to stock on hand and is sold on a first come first serve basis. Call for other great Atari deals!

P.C. EXPRESS 463 Lake Cook Rd., Deerfield, IL 60015

Call Toll Free 1-800-848-0888, Illinois residents call (312) 498-6886

-- MEMORY MOVER --
AND SCREEN SAVE

MEMORY MOVER was written to demonstrate the use of BASIC strings to move massive amounts of memory at machine language speed. Advanced BASIC programmers will find these routines helpful to move player-missile graphics or other data to new locations quickly. Less advanced programmers may use this program to design, save and recall game-board screens and everyone will find this a fun way to doodle on the screen and get some useful insight on how strings are managed in Atari Basic.

This program works on either disk or cassette based systems with 24k or more memories. It can be condensed to fit into 16k machines by removing all of the REM statements, replacing the constants with variables, and using multiple statement lines. A nice feature is the use of a string that has been DIMed to 1 to handle over 4k of memory. This is done by modifying the variable value table after BASIC has performed its space allotment. I used modified strings to move data around in memory and not pay a penalty in DIMed string overhead. Now on to the program.

It is important in this program that SCREEN$ is the first variable on the variable tables. After you have typed in the program, LIST it to disk or cassette and then re-ENTER it. This will clean up the variable tables and put SCREEN$ in the first position and easy to find in the variable value table.

Lines 50 to 220 are the drawing routine. Lines 50 to 110 set up the screen and provide instructions for the program user. In lines 120 through 160, I use subscripted variables to help change the X and Y locations of the cursor. That eliminates a lot of IF-THEN statements and leaves only those needed to check for the edge of the screen. If you modify this program for other graphics modes, don't forget to change the maximum X and Y values. Lines 170 through 210 are used to detect the end of the drawing routine, change color, and provide erasing of lines already drawn. You might wish to add lines to change the color, luminosity, etc. Don't forget to add routines to re-insert those changes after you bring the screen back from storage. Line 220 puts the dots on the screen and loops back.

The flashing routine in lines 250-360 is a graphic way to demonstrate the memory mover technique. Line 250 through 310 provide instructions and reset the graphics mode. After a GRAPHICS command, it is necessary to reposition SCREEN$ to overlay the new screen position. This must be done before you use this string after a GRAPHICS command even if you stayed in the same mode. Lines 320 and 330 look for a signal to end the routine or to change strings.

Lines 340 and 350 do the real work by making SCREEN$ equal to one of the holding tank strings. Instead of moving the screen data pointer, I moved all of the screen data, including the display list. As a program routine, this is not efficient but it is a graphic way of demonstrating the memory mover. When you run the program, you will see that you can swap 4200 bytes of memory fast enough to superimpose two images on the screen at apparently the same time. If you can do it that fast for screen data, you can do it that fast for other data. Line 360 loops you back for more exchanges.

Lines 370 though 510 initalize the program. SIZE sets both holding strings to the correct DIMension for GRAPHICS 7. Room has been allowed for both the display list and screen data. If you modify this program for other graphics modes, don't forget to change the pokes on line 1060-1070, 1100-1110 and 1360 to conform to the new graphics mode memory requirements. Line 430 loads a machine language input/output routine into a string call CIO$. Lines 450 and 460 load the subscripted variables used to read the joystick. Lines 490 to 510 remove all data from the holding tank strings SHOLD1$ and SHOLD2$ and open them up to their DIMed length. If these strings are not opened up, SCREEN$ might send or receive only one byte of data. The string cleaning was thrown in just to show you a fast way of doing it.

Lines 520-610 were added so you could bringing back a screen that you had previously stored without going through the drawing routine.

Lines 620-740 send the program through two draw and save cycles. The variable T stands for test. This was added to show you that the text data is also being moved. Screen data addresses change each time a GRAPHICS command is used. SCREEN$ must be repositioned to conform to the changed data address. This is done by the GOSUB 940. The variables OP1 and OP2 stand

for Operation. OP1 is used to open an I/O control block for input or output operations. OP2 does the same for a CIO location called on by the machine language routine. These variables and WORD$ saved writing different routines for input and output.

Line 750 sends the program to the flashing routine and lines 760-870 call for screen data to be brought in from outside storage and flashed on the screen in the flashing routine. 880 ends the program when you are finished experimenting and 890-920 contain the DATA needed for the CIO$ string machine language routine and the joystick readers.

The heart of this program is in lines 930-1140. BASIC addresses strings and subscripted variables by their offset from the start of the array table instead of their absolute address in memory. To modify a string's address, you must first find out where the array table is. Next, since the strings offset address, current size and DIMensioned size is kept in the variable value table, you must find the value table and the string's position in it. By naming SCREEN$ first in the program, SCREEN$ has the first position on all of the variable tables. The second named variable has the second position and so fourth.

In the variable value table, byte zero and one tell what kind of variable it is and what number variable it is. In string and array variables, byte two and three tell its offset from the start of the array table. In string variables, Byte four and five tell how long it is at a given moment in the program and byte six and seven tell how long it has been DIMensioned to.

I documented this section with wordy variable names and lots of REM statements for easy future refferance. Each string variable repeats this pattern. If you need more than one string mover for future programs, DIM it second (etc.) in the program. That way it will be easy to find on the value table.

Lines 1150 to 1390 are the input/output section. A TRAP statement on line 1170 protects the program from crashing through careless naming of disk files. Lines 1180-1260 get the proper I/O device and a file name if necessary. Line 1270 first closes a file that may have been left open due to error then. Then it OPENs I/O control block one with the code controlled by OP1 and the appropriate name carried by DISKN$. Lines 1280 sets the correct GRAPHICS mode and insures that string SCREEN$ overlays the screen data area. If their is an output operation, line 1290 puts the information to be transmitted on the screen. Line 1310 tells CIO (Central Input/Output, a part of the operating system) what kind of I/O operation will be handled. Lines 1340 tell CIO where to start taking its data. Line 1360 tell how many bytes of data to take and line 1370 call the CIO to action via a short machine language routine. The 16 tells CIO that control block one is being used. Line 1380 CLOSEs the control block and removes the trap so that any other errors in the program can be detected and referanced. If there is an error in I/O operations, the error is TRAPed to lines 1400-1410 which allow you to recover by finding out what is wrong and correcting it without loosing the screen you have drawn.

```
10 REM MEMORY MOVER AND SCREEN SAVER
20 REM
30 DIM SCREEN$(1):GOTO 370
40 REM SCREEN DRAWING ROUTINE
50 GRAPHICS 7:C1=1
60 ? "TEST ";T;" JOYSTICK DRAWS LINE"
70 ? "FIRE BUTTON ERASES DRAWING"
80 ? "SELECT CHANGES COLOR"
90 ? "START BUTTON SAVES DRAWING";
100 X=50:Y=50:PLOT X,Y
110 COLOR C:S=STICK(0)
120 X=X+JOYX(S):Y=Y+JOYY(S)
130 IF X<0 THEN X=0
140 IF X>159 THEN X=159
150 IF Y<0 THEN Y=0
160 IF Y>79 THEN Y=79
170 POKE BUTTON,8:IF PEEK(BUTTON)=6
THEN RETURN
180 IF PEEK(BUTTON)=5 THEN C1=C1+1:FOR
I=1 TO 40:NEXT I
190 IF C1>3 THEN C1=1
200 IF STRIG(0)=1 THEN C=C1
210 IF STRIG(0)=0 THEN C=0
220 PLOT X,Y:GOTO 110
230 REM FLASHING ROUTINE
240 GRAPHICS 0
250 ? "READY FOR FLASHING ROUTINE"
260 ? "THIS ROUTINE MOVES 4,192 BYTES OF"
270 ? "MEMORY AS FAST AS YOU CAN PRESS
THE"
280 ? "FIRE BUTTON. WHEN YOU ARE DONE"
290 ? "EXPERIMENTING, PRESS THE START
BUTTON"
300 ? :? "PRESS RETURN TO START
ROUTINE":INPUT ANSWER$
310 GRAPHICS 7:GOSUB 930
320 POKE BUTTON,8:IF PEEK(BUTTON)=6
THEN RETURN
330 S=STRIG(0)
340 IF S=0 THEN SCREEN$=SHOLD2$
350 IF S=1 THEN SCREEN$=SHOLD1$
360 GOTO 320
370 REM INITIALIZATION
380 SIZE=4192:V256=256:BUTTON=53279
390 DIM SHOLD1$(SIZE),SHOLD2$(SIZE)
400 DIM
JOYX(15),JOYY(15),DISKN$(15),FILENAME$(12)
410 DIM C$(1),ANSWER$(1),CIO$(7),WORD$(10):
C$=CHR$(0)
420 REM BUILD CIO STRING
430 FOR I=1 TO 7:READ
S:CIO$(I,I)=CHR$(S):NEXT I
440 REM LOAD JOYSTICK DATA
450 FOR I=1 TO 15:READ S:JOYX(I)=S
460 READ S:JOYY(I)=S:NEXT I
470 GOSUB 930:REM MODIFY STRING SCREEN
480 REM OPEN UP CLEANED STRINGS
490 SHOLD1$=C$:SHOLD1$(SIZE)=C$
500 SHOLD1$(2,SIZE)=SHOLD1$
510 SHOLD2$=SHOLD1$
520 REM CHOOSE FUNCTION
530 GRAPHICS 0
540 ? :? "DO YOU WANT TO DRAW PICTURES
OR BRING"
550 ? "BACK THOSE YOU HAVE DRAWN?"
560 ? :? "TYPE D FOR DRAW"
570 ? "     B FOR BRING BACK":?
580 ? "THEN TYPE RETURN":?
590 INPUT ANSWER$
600 IF ANSWER$="B" THEN 810
610 IF ANSWER$<>"D" THEN 530
620 REM DRAW 2 PICTURES AND SAVE EACH
630 REM DRAWING 1
640 T=1:GOSUB 50
650 GOSUB 940
660 SHOLD2$=SCREEN$:SHOLD1$=SHOLD2$
670 OP1=8:OP2=11
680 WORD$="SAVE TO"
690 GOSUB 1160
700 REM DRAWING 2
710 T=2:GOSUB 50
720 GOSUB 940
730 SHOLD1$=SCREEN$
740 GOSUB 1160
750 GOSUB 230:REM FLASHING TEST
760 GRAPHICS 0:?
770 ? "ALL SCREEN DATA WILL BE
OVERWRITTEN"
780 ? "BY DATA THAT WILL BE BROUGHT
BACK"
790 ? "FROM THE STORAGE MEDIA"
800 FOR I=1 TO 500:NEXT I
810 WORD$="BRING FROM"
820 OP1=4:OP2=7
830 GOSUB 1160
840 SHOLD2$=SCREEN$
850 GOSUB 1160
860 SHOLD1$=SCREEN$
870 GOSUB 230
880 GRAPHICS 0:? "THIS COMPLETES THE
TEST":END
890 DATA 104,104,104,170,76,86,228
900 DATA 0,0,0,0,0,0,0,0,1,1
910 DATA 1,-1,1,0,0,0,-1,1,-1,-1
920 DATA -1,0,0,0,0,1,0,-1,0,0
930 REM MODIFY ADDRESS OF SCREEN$
940 DISPLAYLIST=PEEK(560)+
V256*PEEK(561)
950 VARIABLETABLE=PEEK(134)+V256*
PEEK(135)
```

```
960 ARRAYTABLE=PEEK(140)+V256*PEEK(141)
970 OFFSET=DISPLAYLIST-ARRAYTABLE
980 HI=INT(OFFSET/V256)
990 LO=OFFSET-V256*HI
1000 REM LOW & HI BYTE OF NEW ADDRESS
1010 REM OF STRING SCREEN$
1020 POKE VARIABLETABLE+2,LO
1030 POKE VARIABLETABLE+3,HI
1040 REM LOW & HI BYTE OF NEW CURRENT
1050 REM SIZE OF STRING SCREEN$
1060 POKE VARIABLETABLE+4,100
1070 POKE VARIABLETABLE+5,16
1080 REM LOW & HI BYTE OF NEW DIMED
1090 REM SIZE OF STRING SCREEN$
1100 POKE VARIABLETABLE+6,100
1110 POKE VARIABLETABLE+7,32
1120 SCREENHI=INT(ADR(SCREEN$)/V256)
1130
SCREENLO=ADR(SCREEN$)-SCREENHI*V256
1140 RETURN
1150 REM NAMING THE OUTPUT FILE
1160 GRAPHICS 0
1170 TRAP 1400
1180 ? :? "DO YOU WISH TO ";WORD$
1190 ? "DISK OR CASSETTE [D/C]?":?
1200 ? "TYPE D FOR DISK OR C FOR
CASSETTE";
1210 INPUT ANSWER$
1220 IF ANSWER$="C" THEN
DISKN$="C:":GOTO 1270
1230 IF ANSWER$<>"D" THEN GOTO 1160
1240 ? :? "TYPE IN THE FILENAME AND
PRESS RETURN"
1250 INPUT FILENAME$
1260 DISKN$="D:":DISKN$(3)=FILENAME$
1270 CLOSE #1:OPEN #1,OP1,0,DISKN$
1280 GRAPHICS 7:GOSUB 940
1290 IF OP1=8 THEN SCREEN$=SHOLD1$
1300 REM SET UP IOCB
1310 POKE 850,OP2
1320 REM LOW BYTE AND HI BYTE OF
1330 REM WHERE TO START MOVING DATA
1340 POKE 852,SCREENLO:POKE 853,SCREENHI
1350 REM LOW & HI BYTE OF HOW MANY
BYTES TO MOVE
1360 POKE 856,100:POKE 857,16
1370 X=USR(ADR(CIO$),16)
1380 CLOSE #1:TRAP 50000
1390 FOR I=1 TO 200:NEXT I:RETURN
1400 ? :? "YOU HAVE MADE AN ERROR"
1410 ? :? "PLEASE TRY AGAIN:GOTO 1160"
```

COMING ATTRACTIONS
August 21st. Meeting

Here's the tentative schedule of topics for the August meeting. Please note that this meeting IS on the 3rd. Tuesday of the month, as will all further meetings this year be.

- Business meeting
- Atari Tennis demo
- Letter Perfect demo
- CBS Educational Software demos
- TARICON '84 workers meeting

Tom Sturza
Program Coordinator

BASIC Subroutines for Filenames

by Phil Heavin

The device independence that is an integral part of the ATARI operating system is a very powerful and useful feature that BASIC programmers do not usually utilize to its fullest extent.

A user of a program that writes an ASCII type output file should have the flexability when he runs the program to direct its output to any device on his computer. This can be achieved very easily on the ATARI by simply prompting the user for a complete file string and using that string in the open statement.

The above approach has two drawbacks. The first is that all users must understand the file naming conventions to give even the simplest response. Second, it would be nice if the majority of the time the user could get what he wants by typing just a RETURN.

The following example program demonstrates the use of the subroutine GETFILE. Given the default file string, DEF$, it will prompt the user for his input, apply the default for each separate portion of the file specification and return the string FILE$ which can be used in the OPEN statement.

```
 100 REM ------------------------
110 REM | PROGRAM TO DEMONSTRATE    |
120 REM | PROMPTING FOR A FILE NAME |
130 REM | WITH DEFAULTS             |
140 REM ------------------------
200 DIM DEF$(16),FILE$(16),DEV$(3),
DFDEV$(3),NAME$(8),DFNAME$(8),EXT$(4),
DFEXT$(4),REST$(12)
210 LET GETFILE=20000
220 FSS=20500
250 REM ------------------------
260 REM | BODY OF PROGRAM GOES HERE |
270 REM ------------------------
300 DEF$="D:OUTFILE.DAT"
310 GOSUB GETFILE
320 PRINT "OPEN FILE ";FILE$
330 PRINT
340 GOTO 310
20000 REM ------------------------
20002 REM |GETFILE ROUTINE          |
20004 REM | PROMPT USER FOR FILENAME|
20010 REM | WITH DEFAULT AND APPLY  |
20020 REM | THEM TO THE RESPONSE    |
20021 REM |                         |
20022 REM |INPUT:                   |
20023 REM | DEF$ - DEFAULT FILE     |
20024 REM |        STRING           |
20025 REM |                         |
20026 REM |OUTPUT:                  |
20027 REM | FILE$ - RESULT FILE     |
20028 REM | DEV$  - RESULT DEVICE   |
20029 REM | NAME$ - RESULT NAME     |
20030 REM | EXT$  - RESULT EXTENTION|
20031 REM ------------------------
20035 FILE$=DEF$
20040 GOSUB FSS
20050 DFDEV$=DEV$:DFNAME$=NAME$:
DFEXT$=EXT$
20060 PRINT "(";DEF$;")";
20070 INPUT FILE$
20080 GOSUB FSS
20090 IF DEV$="" THEN DEV$=DFDEV$
20100 IF NAME$="" THEN NAME$=DFNAME$
20110 IF EXT$="" THEN EXT$=DFEXT$
20120 FILE$=DEV$
20130 FILE$(LEN(FILE$)+1)=NAME$
20140 FILE$(LEN(FILE$)+1)=EXT$
20150 RETURN
20500 REM ------------------------
20510 REM | FSS - FILE STRING SCAN |
20511 REM |                        |
20512 REM |INPUT:                  |
20513 REM | FILE$ - FILE STRING    |
20514 REM |                        |
20515 REM |OUTPUT:                 |
20516 REM | DEV$  - DEVICE         |
20517 REM | NAME$ - FILE NAME      |
20518 REM | EXT$  - FILE EXTENTION |
20520 REM ------------------------
20530 DEV$=""
20540 NAME$=""
20550 EXT$=""
20555 REST$=FILE$
20560 IC=1:LC=LEN(FILE$)
20570 IF IC<=LC THEN IF FILE$(IC,IC)
<>":" THEN IC=IC+1:GOTO 20570
20580 IF IC<=LC THEN DEV$=FILE$(1,IC):
REST$="":IF IC<LC THEN REST$=FILE$(IC+1)
20585 FILE$=REST$
20590 IC=1:LC=LEN(FILE$)
20600 IF IC<=LC THEN IF FILE$(IC,IC)
<>"," THEN IC=IC+1:GOTO 20600
20610 IF IC>1 THEN NAME$=FILE$(1,IC-1)
20620 IF IC<=LC THEN EXT$=FILE$(IC)
20630 RETURN
```

Run this program and try several responses to the prompt. As you will see, typing RETURN would cause the output to be written to "D:OUTFILE.DAT".

If, however, this were just a test run and you respond E: the resultant file string is E:OUTFILE.DAT which means the output would be displayed on the screen.

For printed output you would respond with P:. If you wanted the output to a different file name on disk you would just type the name, for example, MYFILE. The output file will be D:MYFILE.DAT.

If the filename is ok but you are lucky enough to have another disk you can direct the output there by typing D2: causing the file string to be D2:OUTFILE.DAT.

The examples could be continued but rather, try different responses to the sample program. Then use the subroutines in your programs. It will make them more usable for you and those who use your program.

SINCE YOU ASKED ...

By Kathy & Tom Sturza

Q: Why can't M.A.C.E. print a list of the Special Interest Groups (SIG) in each issue of the Journal?

A: We would gladly do so if the Chairpersons of the SIG's would keep us informed of who is in charge and where to contact them. Your current officers haven't had too much success in this area.

However, the following list is the last active list that I have. You may have to track down the new chairpersons.

ASSEMBLER SIG
Todd Meitzner, (313)542-1752

EDUCATION SIG
Mark Davids, (313)774-9709

FORTH SIG
Tom Chrapkiewicz, (313) 772-8291

Not a very long list, is it? There have been other SIG groups in the past. Just to name a few, Telecommunications, BASIC, and Beginners. We've also had suggestions for a PILOT SIG, a LOGO SIG, and a Newsletter SIG.

SIG memberships are FREE but require M.A.C.E. members to organize on their own. M.A.C.E. officers do NOT arrange for SIG's to be created or for meeting locations. Such meetings are usually held in members homes or in local computer stores.

Your officers agree that M.A.C.E. would be a much more worthwhile organization if we had a large number of active SIG's. What do you think?

A COURSE DESCRIPTION FROM THE STUDENT HANDBOOK OF FUNKY WINKERBEAN

COMPUTER MATH 101 -- (PREREQUISITE - Geometry and Algebra I, or eighty thousand plus points on a PacMan machine)

In Computer Math, you'll learn programming, flow charting, numerical base conversions, and how to change the batteries in your pocket calculator.

The language of the computer is called BASIC and you can use it to write programs, solve problems, and communicate with your friends in hyperspace.

Once you've mastered the computer, you should be able to eliminate counting on your fingers once and for all!

CHAOS, MACE and the City of Southfield present

Taricon '84

AN ATARI-ONLY COMPUTER CONVENTION

August 25-26, 1984

SOUTHFIELD CIVIC CENTER PAVILLION
SOUTHFIELD, MICHIGAN

Devoted Exclusively To Atari Home Computers and Related Support Products

CONVENTION FEATURES

* Displays by Software, Peripheral and Accessories Companies
* Product Demonstrations and Workshops
* New Product Introductions
* MANY Exhibit booths with ATARI Related Products
* Interaction with THOUSANDS of Atari Computer Enthusiasts
* Bring the whole Family!

For more information, send for our full pre-registration flyer:

TariCon '84
P O Box 2785
Southfield, MI 48037
or call 313 978 2458

M. A. C. E.

MICHIGAN ATARI COMPUTER ENTHUSIASTS

P.O. Box 2785; Southfield, Mi. 48037

BULLETIN BOARDS: MAIN 978-1685/WEST 582-0657
MACE/TARICON HOTLINE: INFO (voice) 978-2458

NEXT MEETING: 8/21/84; 7:00 PM
Southfield Pavillion--Ten & a Half Mile Road and Evergreen

M.A.C.E.
P.O Box 2785
Southfield, MI 48037

$20.00* FOR A ONE YEAR MEMBERSHIP
(*Payable to M.A.C.E.)

M.A.C.E. MEMBERSHIP APPLICATION

Name _____ Phone _____
Street _____
City _____ State _____ Zip Code _____
Company (if applicable) _____
System Description _____ Disk/Tape _____
Suggestions _____
I can help with . . . _____

( )New ( )Renewal
If a Renewal:
MACE #_____

Date _____ Coupon _____
Amount _____ Membership Card Number _____
☐ Cash Expires _____
☐ Check Number

MICHIGAN ATARI COMPUTER ENTHUSIASTS
P.O. BOX 2785
SOUTHFIELD, MICHIGAN 48037

BULK RATE
U.S. POSTAGE
PAID
PERMIT #431
SOUTHFIELD, MI

IMPORTANT DATED MATERIAL

PLEASE DO NOT DELAY

Printing and Bindery Services by • GRAPHIC ENTERPRISES, INC. Detroit, Michigan • 313-839-6800